DESCRIPTION

DATA STORAGE AND RETRIEVAL APPARATUS AND METHOD OF THE SAME

TECHNICAL FIELD

The present invention relates to a data storage and retrieval apparatus for recording, storing, and reproducing files in a computer or other data processor and to a method for the same.

BACKGROUND ART

Computers and other data processors often use huge volumes of data to achieve desired goals. Ordinarily, this data is classified according to application, type, etc. and the groups of classified data are managed in the form of data files. The data files are stored on hard disks, magneto optic disks (MO), magnetic tape, or other data storage media. When required by the computer or other data processor, the desired data files are read from the data storage medium and used by the applications etc.

Up until now, various types of data storage media and systems for managing these data storage media have been proposed and commercialized. For example, the

data storage system using DTF (digital tape format) standard tape proposed by the assignee is one example. In this DTF tape format, the file size, position information, etc. are recorded at the header part of the tape and the last part of the data stored on the tape to enable high speed file access.

As a system to which the DTF standard is applied, there is PetaServe (name of software offered by assignee, registered trademark). PetaServe is a hierarchical storage management (HSM) software which arranges storage systems in a hierarchy according to the properties of the storage media or data storage capacities so as to enable rational management of large-scale storage systems while improving ease of use as seen from the user's standpoint. PetaServe manages storage systems automatically based on user settings in accordance with the properties of the data storage media, such as hard disks, magneto optic disks (MO), and magnetic tape, and the state of use of the data.

PetaServe for example moves files present on a computer hard disk to a more economical removable medium (DTF tape, magnetic tape, or other storage medium) to secure a large apparent storage capacity of the hard disk. Further, when a need arises for use of

a file moved to the removable medium, the file moved to the removable medium is again restored to the computer hard disk. In PetaServe, files on the hard disk are copied to a removable medium and the original files on the hard disk are compressed. This is called "migration". The restoration of migrated files on to the hard disk is called "reloading".

Figure 7 is a conceptual view of migration and reloading. In Fig. 7, reference numeral 10 represents a computer containing a CPU (central processing unit) 11, memory 12, and hard disk 13 and 20 represents a storage medium comprised of a removable medium. Reference numerals 30 and 40 show the flow of data at the time of migration and reloading. Due to the migration, a file 14 on the hard disk 13 is copied to the removable medium 20, just the information relating to the migrated file is left in the original file, and the rest of the content is discarded. Due to this, a file 14a having the same content as the file 14 stored on the hard disk 13 is stored in the storage medium 20 and the region storing the file 14 on the hard disk 13 can be opened up to other applications. When the CPU 11 etc. desires to use the content of the file 14, it performs reloading to copy the content of the file 14a from the storage medium 20 to the hard disk 13 and

restore the file 15 on the hard disk 13 to its original state, so the CPU 11 etc. can access the restored file 14 by normal file access procedures.

The migration and reloading are performed automatically by PetaServe, so the user does not have to take particular note of it and through processing can be realized.

In the above reloading, however, when restoring a file 14a migrated to the removable medium 20 on to the hard disk 13 of the computer, the content of the file 14a is read from the removable medium 20 and written on the hard disk 13 of the computer 10. Next, the CPU 11 reads the content of the file 14 from the hard disk 13 and stores it in an assigned region of the memory 12. Since the hard disk 13 is interposed in this reloading, the free space of the hard disk 13 is used up and the speed of access to the file 13 as seen from the CPU 11 is limited by the write and read speed of the hard disk so there is the disadvantage that the system performance ends up falling.

Therefore, the present invention was made in consideration of the above problem of the system of the related art and has as its object the provision of a data storage and retrieval apparatus, and a method for the same, which directly writes data read out from

a file stored in a removable medium or other storage medium in a region assigned in a memory of a computer or the like without going through a hard disk when there is a request for access to a migrated file (in an embodiment of the present invention, this being referred to as DDA (direct device access)) and thereby enables the realization of high speed file access without using up the space on the hard disk and without being dependent on the access speed of the hard disk.

DISCLOSURE OF THE INVENTION

The data storage and retrieval apparatus of the present invention is one having a data processor, including a memory, central processing unit, and first storage medium, which performs migration for transferring a file stored in the first storage medium to a second storage medium provided outside of the data processor and then generates an information file including access information for the file in the first storage medium, comprising an information acquisition means for reading said access information from the information file stored in the first storage medium when the data processor accesses a migrated file, a file opening means for opening the transferred file in the second storage medium based on the access

information acquired from the data acquisition means, and a reading means for reading the stored data from the opened file and writing it in a predetermined region on the memory of the data processor.

Further, in the present invention, preferably the data processor is a computer, the first storage medium is a hard disk, and the second storage medium is a removable medium.

Further, in the present invention, preferably the data processor determines a priority of migration based on a predetermined standard for a plurality of files stored on the first storage medium and performs the migration from the file with the highest priority; a file stored on the first storage medium has an information region for storing file management information and a data region for storing data; all of the data of the data region is transferred to the second storage medium by the migration; and an information file is generated on the first storage medium.

Further, in the present invention, preferably, the information file contains the file management information, access information to the file transferred to the second storage medium, and size information of the file on the first storage medium

before the migration, and the data region of the file on the first storage medium is opened up after the information file is generated.

Further, the data storage and retrieval method of the present invention is a data storage and retrieval method where a data processor, including a memory, central processing unit, and first storage medium, performs migration for transferring a file stored in the first storage medium to a second storage medium provided outside of the data processor, then generates an information file including access information to the first storage medium, comprising reading said access information from the information file stored in the first storage medium when the data processor accesses a migrated file, opening the transferred file in the second storage medium based on the read access information, reading the stored data from the opened file, and storing it in a predetermined region on the memory of the data processor.

BRIEF DESCRIPTION OF THE DRAWINGS

The above object and features of the present invention will become clearer from the following description given with reference to the attached drawings, in which:

Fig. 1 is a block diagram of an embodiment of a

data storage and retrieval apparatus of the present invention and shows the configuration of the data storage and retrieval apparatus and the flow of the data at the time of migration and DDA;

Fig. 2A is a view of the configuration of an ordinary file in the present embodiment;

Fig. 2B is a view of the configuration of a bit file in the present embodiment;

Fig. 2C is a view of the configuration of a stub file in the present embodiment;

Fig. 3 is a view of the flow of the migration and DDA;

Fig. 4 is a flow chart of the processing routine for opening a file in the DDA;

Fig. 5 is a flow chart of the processing routine for reading data in the DDA;

Fig. 6 is a flow chart of the processing routine for closing a file in the DDA; and

Fig. 7 is a view of the configuration of a data storage and retrieval apparatus and flow of data at the time of migration and reloading of the related art.

BEST MODE FOR WORKING THE INVENTION

An embodiment of the data storage and retrieval apparatus of the present invention will be explained

next with reference to the block diagram of Fig. 1. The data storage and retrieval apparatus of this embodiment is provided with a data processor including a CPU 11, memory 12, and hard disk 13, for example, a computer 10, and a data storage medium 20 provided outside the computer 10. The data storage medium 20 is for example comprised by a DTF tape, magnetic tape, or other removable medium.

In the data storage and retrieval apparatus of the present embodiment, the CPU 11 operates based on a control program and controls the memory 12, hard disk 13, and other peripherals. The data storage and retrieval apparatus automatically manages the files on the hard disk 13, based on parameters set by the user, according to the properties of the storage media and the state of use of the data, for example, the frequency of use, the time elapsed until use from storage, the transfer and search speed, etc. so as to realize the maximum storage and operational efficiency. For example, it determines the priority of migration based on predetermined conditions such as the frequency of use in the plurality of files stored on the hard disk 13. Further, it performs migration from the file with the highest priority, for example, the file with the lowest frequency of use. Here, as

shown in Fig. 1, it designates the file 14 for the hard disk 13 as the file for migration and performs migration for transferring the file 14 from the hard disk 13 to the external storage medium 20.

The migration in the data storage and retrieval apparatus of the present embodiment is the same as the migration in the above-mentioned PetaServe. First, the content of the file 14 covered by the migration is copied to the external storage medium 20. Due to this, a file 14a of the same content as the file 14 is produced in the external storage medium 20. Further, a stub file storing the access information of the file 14 etc. is produced in the hard disk 13. The stub file stores the access information of the file 14, for example, the bit file ID, file size, etc. and does not hold actual content, so the size becomes far smaller than that of the original file 14. Therefore, the region of storage of the original file 14 in the hard disk 13 is opened up to other applications and the usable size of the hard disk 13 is expanded.

Figures 2A to 2C show the concepts of the original file, bit file, and stub file stored on the hard disk 13.

As shown in Fig. 2A, a file stored on the hard disk is comprised of a header region comprised of an

i-node and a user data region storing the user data. The i-node is formed in byte units, is managed by a file management system, and holds the information required for a file. In the user data region, the user data is stored divided into block units (for example, 512 byte units).

As shown in Fig. 2B, a bit file is comprised of the user data region of the migrated original file and a bit file ID. Note that the bit file ID is an unambiguous ID in the system prepared at the time of migration. The bit file is linked with the stub file by this bit file ID.

As shown in Fig. 2C, a stub file is comprised of the i-node, the first block of the user data region, the bit file ID, the logical size of the file before migration (in both byte and block units), and the stub file ID. The size of the stub file is fixed and is for example set to 1023 bytes.

Due to the migration, a bit file is formed in the external storage medium and only the stub file remains at the hard disk. The user can search through and restore migrated files for access through the stub files remaining on the hard disk. That is, even when a file is migrated, it appears to the user that the original file remains on the hard disk.

The data storage and retrieval apparatus of the present embodiment controls the migration and DDA of the files by an MFS (migrating file system). Note that the MFS spoken of here is comprised of a processor and a control program loaded in the processor. Further, a client in which the function of the MFS is installed is called an MFS client.

Figure 2 is a view of the migration and DDA operation. In Fig. 3, 100 is an MFS client, 110 a data mover (dm), 120 a bit file server (bfsd), 130 a storage server (stsd), 140 a removable media server (RMS), 150 a removable media, and 160 a storage data base. As shown, the removable media server 140 includes a volume server 142 and a library server 144.

The MFS client 100 issues a migration or DDA request and outputs it to the data mover 110 and the bit file server 120. Further, the MFS client 100 provides a TCP stream socket port (data input-output interface) number used for movement of data to the data mover 110 and the bit file server 120.

The bit file server 120 receives the request for migration and DDA from the MFS client 100 and assigns a bit file ID to it. Further, it selects a storage server 130 and a volume server 142 in the removable media server 140. The bit file server 120 manages the

arrangement and information (migration path etc.) of all of the bit files stored in the removable media 150 through the storage data base and outputs the necessary instructions to the storage server 130.

The storage server 130 determines how to store the designated bit file. It determines a suitable volume set for the processing for storage of the bit file requested. Here, the "volume" is one unit of a physical storage unit. For example, in the case of a tape unit, it is one tape. In the case of a unit not allowing simultaneous access of all surfaces (for example, an optical disk unit), it is one surface. "Volume set" means a set of the same type of media. The storage server 130 outputs a request for mounting of a volume or mounting of a disk to the removable storage server 140 or starts up the data mover 110.

The data mover 110 uses the STC stream socket port provided by the MFS client 100 to execute the actual data move from the MFS client 100 to the removable media 150 (or the reverse). When the volume for which movement of data was requested is mounted, the storage server 130 receives the mount information from the removable media server 140 and the data mover 110 is started up.

The removable media server 140 manages all of the

volumes in the removable media 150 and makes requests for mounting or unmounting suitable volumes. Further, the removable media server 140 assigns and opens up volumes to the client and reserves and frees volumes to the client.

The storage data base 160 is a data base (file) managed by the bit file server 120. The information of all of the bit files in the storage is stored in the storage data base 160. The bit file information is updated with every migration or DDA.

Below, an explanation will be made of the flow of migration and DDA in the present embodiment with reference to Fig. 3.

At the time of migration, an instruction signal S100 for performing the migration on a predetermined file on the hard disk is output from the MFS client 100 to the bit file server 120. Further, a TCP stream socket port number S101 used for data movement is output from the MFS client 100 to the data mover 110.

The volume server 142 and storage server 130 are selected by the bit file server 120. Further, the volume server and volume to be used are determined based on the information provided from the MFS client 100, in particular the migration path assigned to the stored file, the size of the file, and other

information. The storage data base 160 is updated by the bit file server 120. Here, when the bit file server 120 is called up by the storage server 130, it assigns a bit file ID to the new bit file which is added to a bit file table in the storage data base 160. Further, the state of use of the selected volume is stored in the storage data base 160. A signal 120 showing the ID of the bit file migrated is returned by the bit file server 120 to the MFS client 100.

Next, a mount request S130 for mounting the selected volume is output from the storage server 130 to the volume server 142 in the removable media server 140. Therefore, the storage server 130 is connected to the volume server 142.

When the selected volume has already been mounted, a signal S140 showing that state is output from the volume server 142 to the storage server 130. The storage server 130 outputs a startup signal S132 to the data mover 110 when it receives information that the selected volume has been mounted from the volume server 142. Further, information such as the socket number of the client, the bit file ID, and the mount point is supplied by the storage server 130 to the data mover 110.

The data mover 110 produces a file in the

selected volume, connects to the MFS client 100, and moves the data. The MFS client 100 reads out the data from the file on the hard disk and writes it to the socket port. The data mover 110 reads out the data from the socket port and writes its in the file produced in the volume.

Due to the operation explained above, the file on the hard disk of the MFS client 100 is migrated and a bit file is formed in the selected volume. A stub file is stored on the hard disk of the MFS client 100 and the data region of the original file is opened up.

The DDA is the reverse to the migration. In the DDA of the present embodiment, the content of a designated bit file is directly stored in a memory region assigned for the application in the client without going through the hard disk of the MFS client 100. Figures 4 to 6 are flow charts of the flow of the file open, data read, and file close routines at the time of DDA. This DDA will be explained in detail below with reference to these flow charts.

The DDA is executed when a request for access to a stub file is issued by an application. For example, when reading the data portion of a file of more than 512 byte size, DDA is executed when there is a write operation in the file or when there is a change in the

size of the file. In this case, a predetermined region of the memory is assigned by the application requesting the access to the stub file and the data read from the accessed file is stored in that assigned region.

Figure 4 is a flow chart of the file open routine. In the MFS client 100, a file open routine starts along with a file open command "dda_open(path, oflag, mode)" from an application etc. Due to this processing, in accordance with a file open command, the bit file descriptor of the file having the path name designated by "path" is opened in accordance with the open flag "oflag". Note that, in Fig. 4, "mode" depicts the mode of the file to be opened.

First, as shown in step S1, the file information of "path" is obtained. When the file is a file which does not exist in the designated migrating file system, an error code is set, the value of "-1" is returned, then the processing is ended (step S12).

Next, as shown in step S2, when the existence of the file designated by "path" is confirmed in the designated migrating file system of the host computer 10, the status of the file in the migrating file system is acquired, then whether the file is a stub file or a shadowed file is confirmed. When the status

the file on the MFS cannot be acquired, an error code is set, the value of "-1" is returned, then the processing is ended. Note that here, a "shadowed file" is a file which exists as both a file on the hard disk in the MFS client and a file of the same content migrated to a predetermined volume.

Then, as shown in step S3, the file designated by "path" is opened. Here, when the designated file cannot be opened, the error code is set, the value of "-1" is returned, then the processing is ended.

Next, as shown in step S4, the bit file information is acquired and whether the media family is a magnetic tape of the DTF format is confirmed. When the media family is not a DTF format magnetic tape, an error code is set, the value of "-1" is returned, then the processing is ended.

Next, as shown in step S5, a link with the volume server is established and the descriptor of the file connected to the link is returned. Here, when the volume server cannot be linked with, an error code is set, the value of "-1" is returned, then the processing is ended.

At step S6, a specific volume and empty drive are reserved from the volume server linked with the file descriptor returned in the previous step S5. When

failing to reserve an empty driver, an error code is set, the value of "-1" is returned, then the processing is ended.

Next, at step S7, the reserved volume, here, the DTF format magnetic tape, is mounted in the reserved driver. When failing to mount, an error code is set, the value of "-1" is returned, then the processing is ended.

After successful mounting of the tape at step S7, as shown in step S8, the volume mounted to the driver is located up to a read position corresponding to the bit file descriptor. Note that, here, when the location to the read position fails, an error code is set, the value of "-1" is returned, then the processing is ended.

Then, as shown in step S9, exactly the prescribed tape buffer size worth of data is read from the read position corresponding to the bit file descriptor to a tape buffer. When failing in the read operation, in the same way as the steps described above, an error code is set, the value of "-1" is returned, then the processing is ended. The tape buffer is provided for example by the removable media server. The data is read from the tape through the tape buffer. At step S10, header information located at the head of the

tape buffer is acquired. Here, when the header information cannot be acquired, an error code is set, the value of "-1" is returned, then the processing is ended.

Finally, when the above series of processing is completely finished normally, as shown in step S11, the file descriptor (fd) opening the file designated by "path" is returned and the processing is ended.

The file descriptor of the file designated by "path" is determined by the processing from step S1 to S11. After that, the data is read from the designated file corresponding to the file descriptor.

Figure 5 is a flow chart of the operating routine when reading data from a designated file. The file is read by reading n number of bytes worth of data from the bit file corresponding to the bit file descriptor "fd" obtained by the above file open routine to the buffer designated by "buf" (here, this is given as a user buffer). Note that, here, the n number of bytes is made a maximum 1 megabyte. When n is 0, "0" is returned and no result is given.

Below, an explanation will be made of the file read operation by referring to Fig. 5.

First, as shown in step SR1, whether the designated size n bytes of the read operation is below

1 megabyte or not is confirmed. When the result is that the n bytes of the designated size is greater than 1 megabyte, an error code is set, the value of "-1" is returned, then the processing is ended.

Next, at step SR2, whether the designated bit file descriptor "fd" is a bit file descriptor obtained by the above file open operation is confirmed. When the bit file descriptor "fd" is not the bit file descriptor obtained by the file open operation, an error code is set, the value of "-1" is returned, then the processing is ended.

Then, as shown in step SR3, the header part at the head of the tape buffer read in the above file open operation is skipped and the pointer is moved to the head of the actual data. Further, the size of the data to be copied to the user buffer is confirmed. Here, the size of the memory copy is n bytes when the actual data read to the tape buffer is larger than n bytes, while the size of the actual data is set when it is smaller than n bytes. Next, the routine proceeds to the next step SR4.

At step SR4, whether the bit file corresponding to the bit file descriptor "fd" is a spanned bit file is confirmed. When the bit file is not spanned, the routine proceeds to step SR10, where exactly the data

of the size set at step SR3 is copied to the user buffer, then the routine proceeds to step SR7. Note that here, a "spanned bit file" means a bit file stored divided between two volumes.

When it is confirmed at step SR4 that the bit file corresponding to the bit file descriptor "fd" is spanned, the routine proceeds to step SR5, where data of exactly the size set at step SR3 is copied to the user buffer, then, when necessary, the next volume is prepared and data read to the tape buffer. Here, when failing in the processing, an error code is set, the value of "-1" is returned, then the processing is ended.

When the processing of step SR5 is carried out normally, the routine proceeds to step SR6, where when there is any remaining area uncopied in the user buffer, data of exactly that amount of bytes is copied from the tape buffer to the user buffer.

As shown in Fig. 5, after either of the processing of step SR6 or step SR10 is carried out, the processing of step SR7 is carried out. At step SR7, when the size of the actual data except the header of the bit file is smaller than the total size of the data copied to the user buffer, that size is returned, then the processing is ended (step SR11).

When the size of the actual data is greater than the total size of the data copied to the user buffer, the routine proceeds to step SR8.

At step SR8, whether there is any data to be copied remaining in the tape buffer is confirmed. When there is data uncopied in the tape buffer, that is, there is actual data remaining in the tape buffer, data of exactly that size is copied to the user buffer. When is no uncopied data in the tape buffer, the routine goes to step SR9.

At step SR9, the data is read from the volume to the tape buffer. When failing in this processing, an error code is set, the value of "-1" is returned, then the processing is ended. When succeeding in reading to the tape buffer, the routine returns to step SR4 and the processing is repeated.

When all of the series of processing succeeds, a non-negative integer indicating the number of bytes of the actually read data is returned. The number of bytes becomes smaller when the number of bytes remaining in the bit file is smaller than "n" and becomes "0" at the end of the file. That is, when "0" is returned as a result of the data read operation, the fact that all of the actual data in the file has been read is shown. Along with this, the data read

operation of the magnetic storage and retrieval apparatus ends.

After the end of the data read operation, the file is closed. Figure 6 shows the file close routine. Below, an explanation will be given referring to Fig. 6.

When closing the file, the bit file corresponding to the bit file descriptor designated by "fd" is closed. Note that here "fd" is a bit file descriptor obtained by the above file open operation.

First, as shown in step SC1, whether the designated bit file descriptor "fd" is a bit file descriptor obtained by the above file open operation is confirmed. When the bit file descriptor "fd" is not the bit file descriptor obtained by the file open operation, an error code is set, the value of "-1" is returned, then the processing is ended. Further, the processing of steps SC2 and SC3 is skipped and the processing of step SC4 is executed.

On the other hand, when the bit file descriptor "fd" is the bit file descriptor obtained by the file open operation, the routine proceeds to the next step SC2.

At step SC2, the specific volume and empty driver reserved at the file open operation are released.

According to this, the reserved volume and the driver are opened up and become available for other processing. Here, when failing in the release of the reserved volume and empty driver, an error code is set, the value of "-1" is returned, then the processing is ended.

After the processing of step SC2 is carried out normally, the routine proceeds to step SC3. At step SC3, the volume server linked in the file opening is released. Here, when failing in the release, an error code is set, the value of "-1" is returned, then the processing is ended.

When succeeding in the release of the linked volume, the routine proceeds to step SC4, where the s "path" designated in file open operation is closed. When failing in closing the "path", an error code is set, the value of "-1" is returned, then the processing is ended.

When the closing of the "path" at step SC4 is carried out normally, at the next step SC5, the device (reserved driver) is closed. Here, in the same way as the steps described above, when failing in the closing, an error code is set, the value of "-1" is returned, then the processing is ended.

When the above series of processing is normally

ended, "0" is returned and the processing ended (step SC6), while when not normally ended, "-1" is returned, so whether the file closing has ended normally or not can be judged by the value returned at the file closing by the application.

As explained above, according to the data storage and retrieval apparatus and method of the present invention, when performing processing for DDA of a migrated file, the content of a bit file stored on a storage medium, for example, a magnetic tape, is read out, then directly written into a memory assigned to an application without going through a hard disk. Therefore, it is possible to avoid using up the space of the hard disk and realize high speed file access without depending on the write speed of the hard disk.

CAPABILITY OF UTILIZATION IN INDUSTRY

The data storage and retrieval apparatus and method of the present invention can be applied to a computer, a computer network comprised of a plurality of computers, or particularly a large-size data processing system handling a large amount of data.